

APPLICATION
FOR
UNITED STATES LETTERS PATENT

TITLE: RETRIEVING DATA FROM A SERVER

APPLICANT: JAMES R. HANSEN

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL624320748US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

Date of Deposit

September 22, 2000

Signature

Samantha Bell
Typed or Printed Name of Person Signing Certificate

RETRIEVING DATA FROM A SERVER

5

Background

This invention relates to a controller embedded in a device (an "embedded controller") that retrieves data from a remote server for a specific instance of the device.

10 A device may contain an embedded controller, such as a microprocessor, to monitor and control its operation. Any type of device may have an embedded controller, including, but not limited to, home appliances, such as washing machines, dishwashers, and televisions, and manufacturing equipment, such as robotics, conveyors and motors.

15 Embedded controllers, also referred to as "embedded devices", are often connected to an internal network, such as a local area network (LAN), with an interface to the Internet. Other devices on the internal network may communicate with the embedded controllers over the internal
20 network. However, the embedded controllers are not generally addressable from the Internet.

Summary

In general, in one aspect, the invention is directed to a controller embedded in a device for retrieving data
5 from a server. The controller sends a command to the server that identifies an instance of the device and receives, from the server and in response to command, data that is specific to the instance of the device.

This aspect of the invention may include one or more
10 of the following. The command may include an operational parameter for the device and the data may include an updated value for the operational parameter. The command may include plural operational parameters for the device and the data may include updated values that differ from
15 current values of the operational parameters.

The data may include a list of operational parameters. In this case, the embedded controller sends a second command to the server, which includes operational parameters from the list, and receives, from the server and
20 in response to second command, updated values of one or more of the operational parameters included in the second command. The data may include a list of operations to be

performed by the controller. In this case, the embedded controller parses the operations from the list and performs the operations from the list.

5 The data may include a configuration file for the device. The command may identify the instance of the device by a device type and/or one or more of a serial number and a universal unique identifier. The embedded controller may send the command to the server periodically. The server may run the Hypertext Transfer Protocol and the
10 command may contain Extensible Markup Language code.

In general, in another aspect, the invention is directed to a server for sending data over a network to a controller embedded in a device. The server receives a command from the embedded controller, identifies an
15 instance of the device from information in the command, retrieves data that is specific to the instance of the device, and sends the data to the embedded controller.

This aspect of the invention may include one or more of the following features. The command may include a
20 device type and/or one or more of a serial number and a universal unique identifier. The instance of the device may be identified based on the device type and/or one or

more of the serial number and the universal unique
identifier. The server may parse the device type and one
or more of the serial number and universal unique
identifier from the command prior to identifying the
5 instance of the device.

The command may include an operational parameter for
the device. The data may include an updated value of the
operational parameter. The data may include a list of
operational parameters for the device. The server receives
10 a second command from the embedded controller, which
includes an operational parameter from the list of
operational parameters, obtains an updated value of the
operational parameter, and sends the updated value of the
operational parameter to the embedded controller.

15 The data may include a list of operations to be
performed by the embedded controller. The data may include
a configuration file for the device. The server may
receive the data specific to the instance of the device and
store the data in memory, from which it is retrieved. The
20 data specific to the instance of the device may be received
via a Web page generated by the server. The server may run
the Hypertext Transfer Protocol and the command may contain

Extensible Markup Language code.

In general, in another aspect, the invention is directed to a system that includes a controller embedded in a device that is capable of communicating over a computer network, and a server that is capable of communicating over the computer network. The embedded controller sends a command to the server over the computer network that identifies an instance of the device and, in response, the server (i) identifies the instance of the device based on the command, (ii) retrieves data that is specific to the instance of the device, and (iii) sends the data to the embedded controller over the computer network.

This aspect of the invention may include one or more of the following features. The embedded controller is not remotely-addressable from the computer network. The computer network is the Internet. The server runs the Hypertext Transfer Protocol and the command may contain Extensible Markup Language code.

Other features and advantages of the invention will become apparent from the following description, including the claims and drawings.

Brief Description of the Drawings

Fig. 1 is a block diagram of a network containing a server and a device having an embedded controller;

Fig 2 is a flowchart showing a process by which the
5 embedded controller retrieves data for the device from the server; and

Fig 3 is a flowchart showing an alternative process by which the embedded controller retrieves data for the device from the server.

Description

Fig. 1 shows a network 10. Network 10 includes a device 11 containing an embedded controller 17. Device 11 is any type of apparatus or system having functions that
10 are monitored and controlled by embedded controller 17.

Device 11 is connected to an internal network 12, such as a LAN. A router or modem 14 couples internal network 12 to an external network 15, such as the Internet/World Wide Web (Web). External network 15 runs TCP/IP (Transmission
15 Control Protocol/Internet Protocol) or some other suitable protocol. Network connections are via Ethernet, telephone line, wireless, or other transmission media.

External network 15 contains a server 19, which is a computer or any other processing device. Server 19 communicates with embedded controller 17 over external network 15 and internal network 12. Embedded controller 17 has a local IP (Internet Protocol) address that can be resolved within internal network 12. However, this local IP address may not be recognizable by devices on external network 15, such as server 19. As such, server 19 may not be able to directly address device 11.

Embedded Controller

Embedded controller 17 runs software 20, which includes web client application 21 and operating software 22. Web client application 21 includes a TCP/IP protocol stack that allows embedded controller 17 to communicate over external network 15. Device operating software 22 provides an interface between Web client application 21 and a database 24. Through device operating software 22, embedded controller 17 retrieves data stored in database 24 and stores data in database 24.

Database 24 is stored in a memory 25 on device 11 or internal to embedded controller 17. Database 24 stores

data, including operational parameters, configuration files, and identification information for device 11.

The operational parameters constitute settings and/or control instructions for the device 11, which are

5 implemented by embedded controller 17. The types of operational parameters that are stored in database 24 depend on the nature of device 11. For example, if device 11 is a heating/cooling system, the operational parameters may include temperature levels, humidity levels, airflow
10 controls, vent/duct open/close controls, and fan motor speed settings. A configuration file is a file that contains a set of one or more operational parameters for an instance of device 11.

What is meant by "instance" is the specific identity
15 of device 11 as distinguished from other identical devices. The identification information stored in database 24 identifies the instance of device 11. This identification information includes, but is not limited to, data identifying the type of the device, a common (or
20 "friendly") name for the device, the manufacturer of the device, the model name of the device, the model number of the device, the serial number of the device, and a

universal unique identifier (UUID) for the device.

The device type specifies a uniform resource locator (URL) for the device, which includes the name of the device. This information identifies a Web site that is
5 associated with, and generated by, server 19 for the device. For example, a device type might be:

www.SonyVideo.com/television/Vega/XBR400

10 for a Sony® Vega® XBR400® television that includes an embedded controller. The common name of the device is how the device is known in the vernacular, e.g., "television". The manufacturer identifies the manufacturer of the device, e.g., Sony®. The model name identifies the particular
15 model of the device, e.g., Vega®. The model number identifies the model number of the device, e.g., XBR400®. The serial number identifies the serial number of a particular instance of the device, e.g., 53266D. The UUID is a universal identifier for the instance of the device,
20 e.g., 4A89EA70-73B4-11d4-80DF-0050DAB7BAC5. Of the data shown above, only the serial number and the UUID are unique to the instance of device 11.

Server

Server 19 is a computer that runs HTTP (Hypertext Transfer Protocol). Server 19 includes a controller 27, such as a microprocessor, for executing software to perform the functions described below. To avoid confusion in terminology, the following reads as though those functions are performed by server 19, even though software in controller 27 of server 19 performs the functions.

Server 19 executes Web server software 29 to communicate over external network 15. Web server software 29 also hosts a Web page associated with device 11. The Web page (not shown) is displayed on the computer of a user, such as the owner of device 11, who may input updated operational parameters for the device. These input updated operational parameters are transmitted to Web server software 29 over external network 15. Web server software 29 stores the updated parameters in database 30.

Web server software 29 stores and retrieves data in database 30 using application logic 32. Application logic 32 is software for accessing database 30 using the CGI (Common Gateway Interface) protocol. CGI is a well-known

protocol for accessing a database. The operational parameters can be stored in database 30 individually or as part of a configuration file for an instance of device 11.

Database 30 is stored in a memory 31, which is inside
5 of, or external to, server 19. Database 30 stores data associated with device 11, including the operational parameters noted above. Other data that may be stored for device 11 is described below.

10 The Data Transfer Process

Embedded controller 17 executes software 20 to retrieve data, such as operational parameters, from remote server 19. Server 19 executes software 34 to send the data to embedded controller 17. Fig. 2 shows these processes in
15 detail. The left half of Fig. 2, titled "Embedded Controller" shows process 40 performed by embedded controller 17, and the right half of Fig. 2, titled, "Server", shows process 41 performed by server 19.

Process 40 generates and sends (201) a command to
20 server 19. The command, or a modified version thereof, is sent by embedded controller 17 to server 19 periodically. It is through this command that embedded controller 17

polls server 19 to determine if there are any new/updated operational parameters for device 11.

5 The command includes data identifying device 11. The data identifies the specific instance of device 11 and includes a device type field and one or both of a device serial number field and a device UUID. The command may also include the common name field, the manufacturer name field, the model name field, and the model number field, as set forth above.

10 The command may be either an HTTP GET command or an HTTP post command. The data included in those commands is similar, with the difference being that the HTTP GET command retrieves a document, such as a configuration file, that contains operational parameters and the HTTP POST
15 command retrieves individual operational parameters. An example of an HTTP GET command is shown in Appendix A and an example of an HTTP POST command is shown in Appendix B.

The HTTP POST and GET commands shown in Appendices A and B contain XML (eXtensible Markup Language) commands.
20 XML is a self-describing computer language in the sense that fields in the XML code identify variables and their values in the XML code. For example, as shown in the

Appendices, the "manufacturer" field identifies a manufacturer, e.g., Sony®, and is delineated by "<manufacturer>" to indicate the start of the field and "</manufacturer>" to indicate the end of the field. XML is used because it can be generated, parsed and read relatively easily by server 19 and embedded controller 17.

As noted, the GET command is used to retrieve a document from server 19. The document to be retrieved corresponds to the fields in the GET command, in particular to the device type, serial number and/or UUID fields. By contrast, the POST command is used to retrieve individual operational parameters. The operational parameters that are to be retrieved are listed in the POST command itself. For example, as shown in Appendix B, the operational parameters include airflow, humidity, motor and vent values for the fictitious "widget" device. The current values of these parameters are specified in the POST command shown in Appendix B as follows:

```
<parameters>
  <Airflow xsd:type="integer">378</Airflow>
  <Humidity xsd:type="double">46.7</Humidity>
  <Motor xsd:type="integer">1500</Motor>
  <Vent xsd:type="integer">4</Vent>
</parameters>
```

The updated values of these parameters are returned by server 19 to embedded controller 17 in a reply POST command. The updated values of these parameters are specified in the POST command shown in Appendix B as follows:

```
<parameters>
  <Motor xsd:type="integer">1250</ Motor >
  <Vent xsd:type="integer">2</Vent>
</parameters>
```

As shown, both the POST and GET commands include the URL of the device in the device type field. As described below, this directs server 19 to a Web site associated with device 11 and, thereafter, in the case of a GET Command, to retrieve a specific Web page that is generated by server 19 for the device. It is noted that, since the POST command retrieves parameters, not a document like the GET command, the POST command need not include a URL of the device.

Referring back to Fig. 2, process 41 (in server 19) receives (202) the command from embedded controller 17.

Process 41 identifies the command as either a POST or GET

command based on a header, such as "POST/CONTROL HTTP/1.1"
(see the headers in Appendices A and B), in the command.
Process 41 uses an XML parser to parse (203) the various
identifying fields, such as device type, serial number, and
5 UUID, from the command.

Process 41 identifies (204) the instance of device 11
based on the information parsed from the command. That is,
process 41 uses the device type, serial number, and UUID
field information to identify the instance.

10
If The Command Is A POST Command

The remaining identification information from the
command is used to narrow the search through database 30
down to data for the specific instance of device 11. The
15 device serial number and/or UUID are used to retrieve
operational parameters specific to device 11.

Once the appropriate data has been identified (204),
process 41 retrieves (205) that data using application
logic 32. Process 41 compares the values of the
20 operational parameters to those included in the POST
command. If the values are the same, process 41 returns an
indication that there are no new/updated values for device

11. If the values of the operational parameters are different, process 41 adds the appropriate updated value fields to the POST command and sends (206) the POST command, with the updated operational parameters, back to
5 embedded controller 17. Thus, only those operational parameters that differ from their original values are returned to embedded controller 17 in the POST command.

If The Command Is A GET Command

10 As was the case above with the POST command, the remaining identification information from the command is used to narrow the search through database 30 down to data for the specific instance of device 11. In particular, the device serial number and/or UUID are used to retrieve (205)
15 a configuration file that is specific to device 11. Process 41 then sends (206) the configuration file to embedded controller 17. The configuration file may be a Web page identified by the URL in the device type field. This Web page is generated by server 19 using parameters
20 stored in database 30 and then sent to device 11. It is noted that the complete Web page itself need not be stored. Alternatively, the GET command may retrieve separate

configuration files and Web pages.

Process 40 in embedded controller 17 receives (207) the data (operational parameters or configuration file) from server 19 in response to sending (201) the command. Process 40 then uses the data to update/reset device 11. For example, if device 11 is a heating system, a new operational parameter may be a new temperature setting for its thermostat. In this example, embedded controller 17 sets the new temperature accordingly. If the device is a television, a new operational parameter may indicate that certain pay television stations are now available. In this case, embedded controller 17 performs any appropriate decoding/descrambling functions on the television signal.

Alternative Embodiment

Fig. 3 shows alternative embodiments of processes 40,41. In processes 40,41, the GET and POST commands request the same parameters each time the commands are issued. The parameters requested are encoded in the software to implement process 40. This embodiment provides a way to change the parameters that are requested without

altering the software that generates the request/command.

Referring to Fig. 3, process 45 in embedded controller 17 begins by sending (301) a command to server 19. The command, in this case, is an HTTP GET command, since it is requesting a document, not individual operational parameters. The document is an XML document that contains a list of operational parameters to be updated. Using this document, embedded controller 17 can change the operational parameters that it periodically updates.

Process 46 in server 19 receives (302) the command from embedded controller 17, parses (303) the command using an XML parser to obtain the information specific to the instance of device 11, and identifies (304) the appropriate document based on this information. As before, the information that identifies the instance of device 11 includes, among other things, the device type, its serial number, and its UUID. Process 46 retrieves (305) the document containing the list of operational parameters to be updated, and sends (306) the document back to embedded controller 17.

Process 45 in embedded controller 17 receives (307) the document from server 19, parses (308) the operational

parameters to be updated from the document, and formulates (309) a POST command to send to server 19. The command is formulated using a command template (not shown), into which process 45 inserts the operational parameters parsed from the document. Process 45 sends this second command to the server. At this point, processes 45 and 46 operate (310) in the same manner as processes 40 and 41, respectively, when used with a POST command. Accordingly, the details of processes 40,41 are not repeated here.

This alternative embodiment may be generalized further. For example, rather than simply retrieving a list of operational parameters, embedded controller 17 may retrieve, from server 19, a list of operations that it is to perform. For example, that list may contain operational parameters to be updated, times at which the updates are to occur, a schedule of diagnostic tests, and the like. Any operation that may be performed by embedded controller 17 may be included on the list.

The process for retrieving the list of operations is identical to processes 45 and 46, save for the contents of the list itself. The actions that embedded controller takes once it has the list (i.e., 310) depend on the

contents of the list. For example, the list might specify that parameters are to be updated every hour and may also contain a list of the parameters to be updated. The list may contain XML commands, which can be parsed by embedded
5 controller 17. Thus, embedded controller 17 reads the commands in the list and performs the appropriate operations with respect to device 11.

Architecture

10 Processes 40,41 and 45,46 are not limited to use with the hardware/software configuration of Fig. 1; they may find applicability in any computing or processing environment. Processes 40,41 and 45,46 may be implemented in hardware (e.g., an ASIC {Application-Specific Integrated
15 Circuit} and/or an FPGA {Field Programmable Gate Array}), software, or a combination of hardware and software.

Processes 40,41 and 45,46 may be implemented using one or more computer programs executing on programmable computers that each includes a processor, a storage medium
20 readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. Also, the programs can be implemented in assembly or machine language. The
5 language may be a compiled or an interpreted language.

Each computer program may be stored on a storage medium or device (e.g., CD-ROM, hard disk, or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the
10 computer when the storage medium or device is read by the computer to perform processes 40,41 and 45,46.

Processes 40,41 and 45,46 may also be implemented as a computer-readable storage medium, configured with a computer program, where, upon execution, instructions in
15 the computer program cause the computer to operate in accordance with processes 40,41 and 45,46.

The invention is not limited to use with the protocols and standards described above. For example, Web server may use Java Servlets, ASP (Active Server Pages), and/or ISAPI
20 (Internet Server Application Programming Interface) to communicate with application logic 32, instead of, or in addition to, CGI. The commands sent by embedded controller

17 and/or server 19 (e.g., in 201, 301, 310) are not limited to HTTP GET and POST commands. Any commands and/or requests for requesting and receiving data may be used.

The data transferred to embedded controller 17 by
5 server 19 is not limited to operational parameters or configuration files. The data may include, for example, a schedule of actions to be performed by device 11 that is based on information pertaining the owner of the device. For example, owner preferences may be stored in database
10 30. The instance-specific data may be used by server 19 to correlate the owner of the device to the appropriate preferences. These preferences then may be transmitted back to device 11 to control the operation thereof.

The original parameters sent by embedded controller 17
15 to server 19 may be used by server 19 to calculate new, updated parameters based on data stored in database 30. Thus, the invention is not limited to simply retrieving updated data, but may also include calculating new data based on currently-available data.

20 The documents and commands described above are not limited to XML format. Any computer language may be used for the commands. The documents may be in any format, for

example, HTML (Hypertext Markup Language) documents may be used. In addition, the invention is not limited to use with the Web, Web servers, and the like. The servers and embedded controllers described herein may be the same type
5 of general-purpose computer appropriately programmed, or different devices.

Other embodiments not described herein are also within the scope of the following claims.

What is claimed is:

10